ANAL68/360
A HIGHER-LEVEL COMPUTER PROGRAMMING LANGUAGE FOR METEOROLOGISTS AND OCEANOGRAPHERS

by

SALOMON FELIX SEROUSSI
B.S., M.I.T.
(1960)

SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE

DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1969

Signature of Author.....
Department of Meteorology, 23 May 1969

Certified by.....
Thesis Supervisor

Accepted by.....
Chairman, Departmental Committee on Graduate Students

Lindgren

ANAL68/360
A HIGHER-LEVEL COMPUTER PROGRAMMING LANGUAGE FOR METEOROLOGISTS AND OCEANOGRAPHERS

by

SALOMON FELIX SEROUSSI

Submitted to the Department of Meteorology on 23 May 1969
in partial fulfillment of the requirements for the degree
of
Master of Science

ABSTRACT

In the past, meteorologists and oceanographers using computers in the solution of their research problems, have been hampered by the highly sophisticated programming languages available. This often required him either to spend a great deal of time learning them or, to rely on a programmer to intervene between him and the computer.

This thesis presents a new higher-level programming language which permits the scientist to interact directly with the computer in his own language in order to solve his problems in meteorology and oceanography.

Thesis Supervisor: Victor P. Starr
Title: Professor of Meteorology

ACKNOWLEDGMENTS

I would like to thank Professor Victor P. Starr for his guidance in the writing of this thesis; Messrs. H. M. Frazier and J. G. Welsh for their indispensable assistance; Dr. Joseph Ercolano and my wife, Selma, for their editing and advice; special thanks to Mrs. Barbara Goodwin and Miss June Ruhwedel for their proofreading and typing of the final manuscript.

TABLE OF CONTENTS

1. Introduction

Meteorological and oceanographic research has always been hampered by the inability of the scientist to deal effectively with the highly complex mathematical equations constituting his physical models, and with the voluminous, non-uniform data collected daily from numerous observation posts throughout the world. To better cope with the "data problem", an international effort is being made which has as its immediate goals the standardization and centralization of all such data collected throughout the world (1).

The question of the highly complex equations appears to be inherent, and constitutes a problem of higher order, which scientists since the time of Richardson have been trying to cope with. The only known effective way of dealing with these equations is via numerical (as opposed to analytical) techniques, although before the advent of computers, such techniques were always quite time-consuming and laborious. The development of large-scale, high-speed scientific computers brought with it the potential to deal with problems where time and labor were important factors. But to use these highly sophisticated

machines with their equally sophisticated control languages and operating systems presupposes extensive knowledge and experience on the part of the user, and hence requires an intermediary (a computer programmer who often does not have the sufficient specialized background) to intervene between the computer and the scientist, in a language the latter frequently does not understand.

The purpose of this thesis is to describe a higher-level computer language which permits the scientist to interact directly with the computer in his own language in order to solve his problems in meteorology and oceanography. The first such techniques were developed for the UNIVAC 1108 by James G. Welsh of the Travelers Research Center, Inc. (2). In this thesis the problem has been completely reworked. The new results have been arranged for the IBM 360 family of computers by the writer. This includes the largest currently available member of this family, the IBM360/91.

2. Higher-Level Programming Languages

The use of higher-level programming languages to provide computer accessibility to non-professional programmers is not new. Their evolution was in direct response to the scientist's needs. Systems like ICES (Integrated Civil Engineering System) (3), GPL (General Processor Language) (4), LISP (5), SNOBOL (6), APL, and many others provide different scientific and engineering disciplines with access to computers in their own language.

The process by which these systems were developed seems to have followed a pattern wherein special systems were replaced by more general ones (7). These general programming systems were first developed as a result of the need to avoid repetition in the coding of programs. Later on, the availability of systems such as FORTRAN made it possible for the professional programmer to code new problems with relative ease. It also allowed him to look into the possibility of attaching to languages such as FORTRAN special commands that could be used by the scientist to formulate problems in his own language. The result of these changes were the special purpose higher-

level programming languages. These systems relieve the scientist of almost all problems related to the computer and allow him to concentrate on the solution of his problems.

ANAL68/360 is a special purpose higher-level processor oriented to the solution of meteorological and oceanographic problems. The routines which form the basis of ANAL68/360 were mostly written in FORTRAN under a time shared system which provided for online checkout and debugging.

3. Meteorological Research and Computers

Meteorology is the study of man's oldest scientific problem: the understanding and prediction of the behavior of the earth's atmosphere.

The fundamental laws of physics have long been understood. Their application to the motion of the atmosphere involves equations that are so highly non-linear that there is no hope of finding analytic solutions. In an attempt to find a solution to these equations, numerical techniques have been developed which require an enormous amount of calculating. The first attempt at applying these techniques was done by Richardson

(1922), and it showed the magnitude of the task when a day's forecast took him one year to complete (8).

Further work in the theoretical field established the methods which are today's basis for the study of the general circulation of the atmosphere:

1) Numerical weather prediction as described above.

2) Statistical weather prediction, i.e., the application of the theory of stationary time series.

The development of high speed computers permitted a new approach to these problems by allowing an increase in the size of the models used to simulate the behavior of the atmosphere. Moreover, the use of computers helped in the solution of another related problem: the amount of data that is gathered daily around the globe and which is used by the scientist as input to his models. Some earlier studies were based on data gathered at a particular station; others involved hemispheric research describing the distribution of several parameters. When attempting to summarize all these studies and combine them into a single picture, serious problems were encountered. First, the data used spanned different periods of time and space and as such were incongruentwith each other. Second, theoretical inconsistencies

were encountered in the formulation of the studies (9).

In search of a solution to these problems, the National Science Foundation undertook the sponsorship of a comprehensive observational study under the direction of the Planetary Circulations Project at M. I. T. with the assistance of the Travelers Research Center, Inc., Hartford, Connecticut (10). The first step in this project was the compilation of five years of observational data under the Input Data Handling phase of the program. The period processed was from May 1958 to April 1963 for observations taken at 0000 GMT at 704 stations located over the northern hemisphere. Incorporated into this data bank were observations extending into the southern hemisphere and at 1200 GMT for the North American stations (11). The data base being available, a series of comprehensive studies are being conducted as part of the second phase of the work being done at M. I. T. ANAL68/360 is an effort to provide the scientist better tools with which to perform this work.

4. ANAL68/360 - Basic Concepts

4.1 Introduction

The development and analysis of meteorological problems has shown a commonality of the computational routines used to solve them. These common attributes are such that problems differ from one another no so much in newly conceived routines but in the manner in which existing programs are reordered in sequence.

This suggests that an effective computer program can be written to assist in the solution of meteorological problems by identifying the commonly used computational routines and making them available to the scientist in any sequence he desires. In other words, if we can assemble a set of conceptual building blocks with which the scientist can formulate his problems, the solution to any specific problem is merely to devise the sequence for connecting these building blocks into the necessary procedure. These building blocks are called operators. The suggested computer program can then be described as a library of operators with a convenient means for specifying the usage sequence.

The successful implementation of this program (we call it ANAL68/360) depends on the ability to identify an operator

set both comprehensive and meaningful to the scientist and to design an overall program structure which provides a standard, workable interface to each operator. The operator set is effective because it contains not only many two-dimensional, field-oriented mathematical operators familiar to the scientist, but also less familiar operators to facilitate problem structuring, objective analysis, mapping transformation, and graphical output. The program structure is effective because it is based on manipulating two-dimensional fields defined by a standard set of attributes.

The principal advantage of ANAL68/360 is that it pre-defines the computational building blocks with which to formulate a problem. It is our experience that this is an area that is not widely appreciated by scientists, who focus on the computational formulas required by a problem, neglecting the relationship of the mathematical formulas to the less well-defined organizational interactions of the problem. These more obscure aspects of every problem and their contribution to the overall organization are crucial to realize an effective computer program. Many ANAL68/360 operators implement clear-cut mathematical formulas, but there is also a group of operators which are useful

in defining less obvious needs of any problem. Adopting the concept implemented in ANAL68/360 has the advantages that (a) problems can be conceived in terms of pre-defined, immediately-available operators and (b) newly required operators can be easily implemented because of the standard program interface for each operator.

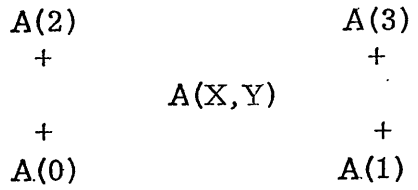
4.2 Fields

Because most familiar operators perform computations on two-dimensional arrays, the basic organizational element in ANAL 68/360 is a field. A field consists of (a) a six-character name, (b) parameters which define the areal extent of the field, and (c) a rectangular array of grid point values, which defines the spatial distribution of the field.

The field itself is represented by an array of gridpoint values. Most operators are written for finite-differencing differential schemes, in which case, the grid matrix is usable as it stands. Operators that require a continuous definition of the field, use a simple interpolation formula to provide continuous values between gridpoints. The interpolation formula is

$$A(X, Y) = A(0) + X[A(1)-A(0)] + Y[A(2)-A(0)] + XY[A(0) + A(3) - A(1)-A(2)]$$

where notation is explained below.



$$0 \leq (X, Y) \leq 1$$

The areal extent of that field is defined by six parameters.

(A) The rotation of the field grid from some reference.

(B) The mesh size of the field grid.

(C) The X-coordinate of the grid rotation pole.

(D) The Y-coordinate of the grid rotation pole.

(E) The number of gridpoints in the X-direction.

(F) The number of gridpoints in the Y-direction.

Strictly speaking, these grid specifications only allow the relative conversion of a field with one extent to some other extent, but in practice, the grid specifiers usually relate the grid-space field to some geographical extent on the earth's surface. For this case, the grid rotation pole is either the North Pole or the South Pole. The rotation is that of the grid Y-axis westward from Greenwich, and the grid mesh is the ratio between the JNWP grid and the field grid, such that finer grids have a mesh greater than 1.0. Since we have tied ourselves to the JNWP grid via the

grid mesh, we are also adopting the polar stereographic projection true at 60 deg. N or 60 deg. S as our mapping projection. The specification of a field grid is described more fully in the following section.

All fields are named in ANAL68/360, and certain names are recognized to have special significance. Names such as STREAM, POTNTL, VRTCTY, DVRGNC, LAPLAC, U, V, DIRECN, and MAGNTD will probably be easily recognized by the scientist as well as by ANAL68/360. The latter four names are peculiar to vector fields, and except MAGNTD, denote vectors whose reference coordinate system is the local terrestrial system (East, North system). When a vector is oriented with respect to the grid system (X, Y system), comparable names are UO, VO, DIRECO. U and V (or UO and VO) are interpreted by ANAL68/360 to be the components of any two-dimensional vector, not velocity only. To illustrate how ANAL68/360 will use these field names, consider the following examples.

(1) When the Laplacian operator ∇^2 is applied to a field named STREAM or POTNTL, the resulting field is named VRTCTY or DVRGNC, respectively.

(2) When a vector (U, V) is converted to direction-and-speed fields, the field names are DIRECN and MAGNTD.

(3) When a vector (U, V) is converted from terrestrial reference to grid reference, it is renamed (UO, VO).

The effectiveness of ANAL68/360 lies in the ability of the scientist to formulate his problems directly in terms of fields, rather than individual elements, such as grid-point values or computer words. Field manipulations are carried out by field-oriented operators and by the internal bookkeeping of ANAL68/360. The great flexibility of ANAL68/360 means that the full repertoire of operators is available for even the simplest problem. This is especially useful in the area of graphical output. The ease of getting a problem on the computer means that many simple "single-shot" experiments are now practical, and what once were simple problems can easily be developed into elaborate, but meaningful experiments which can yield far more sophisticated results.

The limitations of ANAL68/360, in terms of fields, are related to the computer state-of-the-art. The maximum number of fields that can be simultaneously contained in the computer is six. Each field may consist of an array of grid-points with maximum dimensions 47 x 51 gridpoints. (This is again related to the

JNWP grid, for it is the rectangle which contains the JNWP octagonal grid.) Since most operators use only two or three operators at a time, six arrays in the computer is usually adequate. For problems requiring more than six, operators provide for efficient transmission of fields to and from external storage devices, effectively increasing the field capacity of ANAL68/360, to typically, 120 fields.

4.3 Grids

The standard mapping grid for meteorological problems was defined by the Joint Numerical Weather Prediction unit in the 1950's. The grid was based on the computer limitations and the plotting map commonly in use. The grid-points lie with 1-inch spacing on a polar stereographic map, 1:15 million true at 60 deg. N. A rectangle 47 x 51 gridpoints is defined, with the lower left point assigned (0, 0), the North Pole assigned (23, 25), and the J-direction of the grid aligned along 80 deg. W. No points were allowed in the corners of the grid, giving it an octagonal shape (these corners in the computer are available for instructions).

A more useful grid specification is allowed by generalizing the JNWP grid. First, the grid can be subdivided A-fold

about the origin (0, 0); this relocates the North Pole at (23A, 25A), for a subdivided JNWP grid. Note that A need not be an integer value, and can be even less than 1, resulting in a grid mesh coarser than the JNWP. Second, the Y-axis of the grid can be aligned with any arbitrary longitude R, where longitude is measured positive westward from Greenwich. Third, the North Pole can be assigned arbitrary coordinates [X(P)A, Y(P)A], where [X(P), Y(P)] are the coordinates of the pole on the unity-mesh grid with the same orientation.

The parameters A, R, X(P), and Y(P) define an infinite grid system. A finite sub-grid on which fields can be evaluated must be selected from the infinite grid. Thus the rectangular grid ($I1 \leq I \leq I2$, $J1 \leq J \leq J2$) is specified by the limits I1, I2, J1, and J2, and of course, the number of points in each direction is not allowed to exceed the capacity of the program, viz, 47 x 51.

One last generalization of the JNWP grid involves the specification of a comparable grid system in the southern hemisphere. The convention is adopted that the values of A prefixed by a minus sign will signify a grid centered on the South Pole, for a polar stereographic map projection true at 60 deg. S.

Furthermore, the grid rotation R will refer to the -J-axis.

An ANAL68/360 grid is characterized by the eight parameters R, A, X(P), Y(P), I1, J1, I2, and J2. This is called the external definition of the grid. ANAL68/360 converts these to an internal definition of six parameters, ROTN, AGRID, XPOLE, YPOLE, ILONG, and JLONG, as follows.

$$\text{ROTN} = R, \text{ and AGRID} = A$$

$$\text{XPOLE} = 1 + X(P) \left| \text{AGRID} \right| - I1$$

$$\text{YPOLE} = 1 + Y(P) \left| \text{AGRID} \right| - J1$$

$$\text{ILONG} = 1 + I2 - I1$$

$$\text{JLONG} = 1 + J2 - J1$$

The following formulas are also found to be useful.

$$XP = (\text{XPOLE} + I1 - 1) \left| \text{AGRID} \right|$$

$$YP = (\text{YPOLE} + J1 - 1) \left| \text{AGRID} \right|$$

Note that the internal grid specification is merely a redefinition of the external coordinates such that the lower left hand grid-point is assigned the coordinates (1, 1), and the pole position is given on this translated grid.

With this development of grid notation so related to map projection, one must be aware of the fact that, by and large,

ANAL68/360 usually operates in the pure grid space, and is completely unaware of any reference to location on the surface of the earth. Only a handful of operators (VEARTH, VGRID, LATLON, MAPFAC, and SINLAT) provide the required earth-grid referencing. This means differential operators are computing the change per grid-interval, and not per meter or nautical mile on the earth's surface. The conversion between grid-space-differentials and terrestrial-differentials is effected by use of the operators VEARTH, VGRID, and MAPFAC.

4.4 Observational Data

In addition to manipulating two-dimensional fields, ANAL68/360 contains operators to allow the generation of fields from randomly spaced observational data. This process is called objective analysis. The nature of observational data as opposed to fields and the general method of handling this data is as follows.

A field is a representation of some physical variable on a regular, convenient grid. Observational data, on the other hand, is not regularly spaced, and usually includes values of several physical variable ϕ's at the observing station. In

ANAL68/360, observational data is organized as an array of stations, each station comprised of an X coordinate, a Y coordinate, and one or four observed values. If there is only one observed value at each station, as many as 1598 stations can be handled in the computer. With four ϕ's at each station, as many as 799 stations can be input.

Two of the six ANAL68/360 field arrays A(5) and A(6) are used to hold observational data. For data with one ϕ and fewer than 800 stations, only A(5) is required, and the the sixth array is available for field use. Otherwise both A(5) and A(6) are required to hold observational data. Obviously, an array holding data cannot be used for field computations without destroying the observational data.

The stations in the computer can be divided into as many as 40 blocks by inserting block boundaries in the list of stations. Most operators manipulating data handle stations by blocks rather than by individual stations, or by the total data list. This allows meaningful testing of objective analysis techniques, when certain blocks of observations are withheld from the analysis technique for comparative evaluation. Data blocking

also allows stratifying observations by quality or some other attribute.

5. Programming Concepts of ANAL68/360

5.1 ANAL68/360 as a Hyper-Processor Simulator

The programming concept fundamental to ANAL68/360 is to define a "hyper-processor" for meteorological problems, and then to code a program which simulates the hyper-processor on any specific computer. The memory of this processor should be organized into two-dimensional arrays, rather than individual words or elements. The instruction repertoire should include sophisticated, array-oriented operators, that can be sequenced to solve a wide range of problems.

When an ANAL68/360 simulator is written for any particular computer, the storage capacity of the specfic computer is divided into three areas: (1) the ANAL68/360 arrays, (2) ANAL68/360 bookkeeping, and (3) the ANAL68/360 simulator code. For present computers, six ANAL68/360 arrays, each 47 x 51 (to accomodate the JNWP grid) have been adequate, while leaving sufficient storage for the bookkeeping and an effecient

portion of the simulator. The bookkeeping section of storage is on the order of 1000 to 2000 entries, for keeping track of array contents and the sequence of hyper-instructions. The remainder of the computer storage will contain ANAL68/360 simulator code; if this storage is insufficient for the entire simulator, overlaying devises are usually available to allow the simulator to be partioned to segments that can fit in the available storage.

All computer systems adequate for ANAL68/360 include input-output devices that can be used for auxilitary data storage. Magnetic tapes have been long used for this purpose, and in ANAL68/360 provision is made for reading and writing fields and observational data on magnetic tape. A second type of storage device, such as magnetic drum and disk storage, are termed "random-access" because the access to any part is limited only by relatively fast mechanical access mechanisms, whereas the access to any element on a magnetic tape can involve long delays for moving the tape medium. The availabilty of random-access mass storage allows a considerable enhancement of the effectiveness of ANAL68/360.

The essential concept for utilizing drum or disk storage, is to create a large number of two-dimensional arrays (typically,

120 arrays) similar to the six arrays in the computer storage, and to provide an efficient means for transferring fields between the computer arrays and the external drum or disk arrays. This redefines the structure of the ANAL68/360 hyper-processor to consist of a "central-processor" which includes six working arrays and an "array-store" on the random-access external device. This has been found to be a powerful technique for utilizing drum and disk storage.

5.2 ANAL68/360 Statements as Hyper-Processor "Machine Language"

The input to the ANAL68/360 hyper-processor consists of the procdure of operators to be executed, along with any fields and observational data required by the procedure. The cards describing and ANAL68/360 procedure are essentially the hyper-machine language for programming the ANAL68/360 hyper-processor. The purpose of this section is to describe the ANAL68/360 statement format.

Each ANAL68/360 statement requires one or more punched cards (see Figure 1). The six-character name of the operator appears in columns 1-6 of the statement card. Also on the statement card(s) are punched the numeric values or

alphabetic names required by the operator. As many as four six-character names can be punched in columns 7-30 of each statement card, or as many as six 7-column numeric values can be punched in columns 39-80. No operator can require both alphabetic name and numeric values. An operator, which requires more than four names or six numeric values will use as many consecutive cards as necessary, but each subsequent card contains ETCbbb in columns 1-6, instead of the operator name (where b=blank).

The names or values required by an operator are called arguments. In addition to these arguments, each operator allows three further descriptors. Some sort of simple option, and two integer values, called N1 and N2. In general, N1 and N2 are used to indicate which fields are to be operated upon, and the option determine which of two possible paths should be taken with the operator. The values for N1 and N2 are punched in columns 31-34 and 35-38 respectively. The option is indicated by the presence of blanks (option true) or any punching (option false) in columns 25-30, since these columns are also used for the fourth alphabetic argument, no operator with four or more name-type arguments is allowed an option.

The operator write-ups include, along with a general description of each operator, the significance of N1, N2, option, and any arguments required by the operator. Note than an operator may not require any of these descriptors.

5.3 ANAL68/360 Operators

The following groups of ANAL68/360 operators include a list with brief descriptions of most operators. The operator groups provide:

Mathematical Manipulations of Two-Dimensional Fields

ADD	Adds or subtracts two fields
AVERAG	Average value and squared value of a field
ATAN	Radian arctangent of the quotient of two fields
COS	Cosine of a field in radians
DD/FF	Converts a vector from components to direction, speed
DIVERG	Computes divergence of a vector
DI	Fundamental differencing operators for
DJ	backward (DBI and DBJ) and forward
DBI	(DI and DJ), non-centered differences
DBJ	
EXP	Computes "e to the field power"

GRADIE	Computes the gradient vector of a field
JACOBA	Jacobian formulated for non-linear problems
JACOBX	Defined $(\partial A/\partial X)\ (\partial B/\partial X) + (\partial A/\partial Y)\ (\partial B/\partial Y)$
JACOB1	Simple Jacobian of two fields
-KX	Rotates a vector 90 degrees counter-clockwise
LAPLAC	Computes the Laplacian of a field
LOG	Log-to-the-base-e of a field
LOOKUP	Generates a field from other fields by table look-up
MOVE	Converts a field on one grid to another grid
MULTIP	Multiplies or divides two fields
PLANE	Generates a field that is a plane
POISSO	Computes the inverse of the Laplacian
PTAGE	Generates a grid of 1's and 0's from a field
SIN	Sine of a field in radians
SMOOTH	Smooths a field
SQRT	Computes the square root of a field
SUMI	Non-centered, finite-difference
SUMJ	algorithms for indefinite
SUMBI	integration
SUMBJ	

TAN	Tangent of a field in radians
TXFORM	Scalar transformation of a field
YV	Converts a vector to components from direction, speed
VORTIC	Computes the vorticity of a vector
ZERO	Generates a field of zeros

Manipulation of Observational Data

DIFFER	Average difference between observations and a field
DISCAR	Erases observations with large differences from a field
EXTREM	Generates observations at field extrema
INTERP	Interpolates observations from a field
DHIDIV	Divides or multiplies two observed parameters
PHISUB	Subtracts or adds two observed parameters
PHIU, V	Converts observed direction, speed to components
TRANSL	Displaces observations in a flow field

Utilization of External Storage and Input-Output Devices

DATA	Reads formatted observations on cards
FETCH	Transmits a field from drum to core storage
FIELD	Reads a formatted field on cards
LIST	Prints a set of observations
PRINT	Prints the grid values of one or two fields
PUNCH	Punches a field on cards

PUNCH3 Punches a field in EBCDIC codes

READ Reads a field or observation set in ANAL68/360 format

SAVE Transmits a field from core to drum

WRITE Writes a field or observation set in ANAL68/360 format

Graphical Output

BANDS Sets print character set for shaded contour bands

SHADE Prints a field with shaded contour bands

4020ST Starts a new microfilm frame

4020EX Labels field extrema on a microfilm frame

4020PL Isopleths a field on a microfilm frame

4020DA Plots observations on a microfilm frame

Convenience for the Polar-Stereographic Map Projection

GRID Specifies the analysis grid space

LATLON Generates latitude, longitude fields

MAPFAC Multiplies fields by the polar-stereographic map factor

SINLAT Computes sine of latitude for polar stereographic grids

VEARTH Converts a grid-referenced vector to earth reference

VGRID Converts an earth-referenced vector to grid reference

Analysis Control and Specification

END Marks the end of a program loop

EXECUT Marks the end of an ANAL68/360

GRID Specifies the analysis grid space

LOOP Marks the beginning of a program loop

PLINES Sets print control to the appropriate lines/inch

6. ANAL68/360 Programs

6.1 Introduction

ANAL68/360 is a programming system for the solution of meteorological and oceanographic problems. It consists of (1) a language that describes the problem and (2) a processor (computer program) that accepts this language and and generates the requested results.

The language description and the philosophy behind it were defined in the previous section. It is the scope of this section to explain the reasons why ANAL68/360 was implemented and how it was done.

Since its inception in the early 1960's, ANAL 68/360 and its predecessors have been utilized by a relatively small numer of scientists. There are several reasons behind this fact but one of the most important is that the programs were implemented for large scale computers, and thus not accessible to groups with limited resources. By providing fully compatible

programs for the IBM 360 series it will be possible for users with access to a small machine, 128K minimum, running either 360/DOS or OS, to set up the procedures to be run under the diagnostic mode where execution speed is not critical since they take a few seconds to run. This facilitates the debugging of the procedures.

Once a program has been checked out under diagnostic mode, then it can be executed on a larger machine, be it another IBM 360 or a UNIVAC 1108. It is also conceivable that a user with access to a large 360 might want to execute his programs on an 1108 so as to get his output on the 4020 microfilm recorder.

ANAL68/360 is also available under the IBM 360/67 Time-Sharing System, making it accessible, on an experimental basis, to any user with an IBM 2741, switched typewriter terminal, or IBM 1050 terminal.

6.2 Description of Programs - Program Flow

The programs written for the IBM 360 are those which by necessity must be in assembly language and are thus not compatible in instruction format with those available for the UNIVAC 1108. A typical program flow is shown in Figure 2 to give a basic

understanding of the steps taken by ANAL68/360 in response to user requests.

It is the task of the main routine, ANAL360, to direct the flow of information as requested by the control cards prepared by the scientist. The main steps taken are:

STEP 1: Job control cards indicate to the operating system the program to be run and its input-output and core requirements. The supervisor in the operating system will pass control to the main routine in the ANAL68/360 program at its entry point, ANAL360, which will then initialize certain variables.

STEP 2: ANAL360 will pass control to a FORTRAN subroutine, A8EXEC.

STEP 3: A8EXEC will call a subprogram, A8INIT, to initialize counters and input areas, returning control when done.

STEP 4: A8EXEC reads in the control cards containing the procedures written by the user. Figure 3 shows a coding form available for this purpose. The ANAL68/360 operator is read with the FORTRAN FORMAT (5A6, 2F4.0, 6E7.0) from pre-punched cards containing information as described in section 6.2.

STEP 5: As read in, each operator is printed with a format (5A6, 2F5.0, 1PE12.7). Both input and output formats differ slightly from the ones used in the UNIVAC 1108 programs, but the differences are minor and do not affect the results. Figure 4 is an example of the output provided by ANAL68/360.

STEP 6: For each operator, one or more entries are made into the path table in the format shown in Figure 5. An example of a path table is appended to the procedure shown in Figure 4.

STEP 7: When A8EXEC detects any of the ending operators, EXECUT, EXIT, INITIA, or it reaches a maximum of 512 entries in the path table, it will terminate reading control cards and return control to the main routine (unless serious errors have occurred).

STEP 8: The main routine will then proceed to analyze the entries in the path table.

STEP 9: From the value provided in the path entry for the relative position of the operator from the beginning of the table, the main routine can obtain the values to check against the input parameters.

STEP 10: The checking is done for all entries in STABLE, provided the operator is not EXECUT in which case return will be made to STEP 2 for execution of the next procedure. If an error is found while checking, the error routine, A8MESG, will be called to print the proper message.

STEP 11: In diagnostic mode, processing will always continue by making a correction to the value in error. Otherwise the procedure is ended and we return to STEP 2.

STEP 12: If the routine requires a subprogram to be executed, control will be passed to it.

STEP 13: On return to the main routine, posting of results, if required, will be made.

STEP 14: Processing of the next pathentry is initiated.

STEP 15: If the EXECUT operator is detected, it indicates the end of the procedure. Control is passed to A8EXEC to read the next procedure. If an empty card reader is detected, the job will be considered finished and control reverts to the operating system.

Appendix I is a description of the parameters used by ANAL68/360 (located in name COMMON, A8CMIS). Appendix II gives some examples of the operators available and the parameters

they require.

6.3 Error Diagnostics

Error messages are provided by both the main routine and by each subprogram called to execute a particular operator. Assembling a procedure will not be stopped by a bad statement. A procedure will start with the first recognizable control card, and ends with one of the three cards, EXECUT, EXIT, or INITIA. The ANAL 68/360 executive, which processes statements, may generate the following messages:

1. EXECUTIVE READING STEP XXX CANNOT RECOGNIZE CARD STARTING ...

   Columns 1-6 of the indicated card do not correspond to a recognizable operator or one of the three procedure-ending statements. A special "illegal" operator is generated in its place, which will terminate execution mode, but is ignored in diagnostic mode.

2. EXECUTIVE READING STEP XXX ATTEMPTING TO ACTIVATE MORE THAN SEVEN COUNTERS.

   Eight more START (or LOOP) statements than TEST (or END) have been encountered. An "illegal" operator is generated as in 1.

3. EXECUTIVE READING STEP XXX ATTEMPTING TO DE-ACTIVATE MORE THAN SEVEN COUNTERS.

   More TEST (or END) statements than START (or LOOP) statements have been encountered. An "illegal" operator is generated as in 1.

4. EXECUTIVE READING STEP XXX EXCEEDS MAX PROCEDURE WITH CARD STARTING ....

   The statement which would exceed the maximum allowable procedure length is listed. The execution is attempted up to, but not including, the listed statement, but the listed statement is retained for processing after execution. Attempted execution may encounter two more errors: first, error (6) below, if TEST statements remain unprocessed, and second, if the executed procedure required card input, those data cards will not be properly positioned in the card deck, because they are still preceded by the unprocessed statements of the procedure.

5. EXECUTIVE READING STEP XXX EXPECTED ETC CARDS STARTS ....

   An operator that requires more than 4 alphanumeric or

6 floating-point arguments, is not followed by sufficient ETC cards. The unspecified (remaining) arguments will be zero , and the non-ETC is processed as the next statement.

6. EXECUTIVE AFTER STEP XXX TRYING TO EXECUTE A PROCEDURE WITH X COUNTERS NOT ACTIVATED. DIAGNOSTIC MODE NOW.

All procedures must contain equal number of START (or LOOP) and TEST (or END) statements for proper looping. Execution mode of any procedure not satisfying this condition is not allowed.

6.4 Entering Diagnostic Mode

Except in the case of error 6. above, the ANAL68/360 executive always starts a procedure in execution mode. The operator DIAGNO within the procedure sets diagnostic mode for the rest of the procedure. In diagnostic mode, the following operators are fully effective: ALTER, COUNTS, ENTER, NAME, PROCEDURE, STATUS, START, TEST, VALUE, GRID, ATABLE, FTABLE, LOOP, and END. The following operators, the results of which depend upon input data, react as follows:

READ: for N1 = 0, a data load with one ϕ and no stations is assumed.

for N1 = 1-6, a field with name READ on the current analysis grid is assumed. No actual input.

DATA: A data block with one ϕ and no stations is assumed. No actual card input.

All other operators behave logically correct in diagnostic mode, but no computation, input, or output is performed. This means that diagnostic mode will run several hundred times faster than execution mode, permitting an inexpensive, complete exercising of a procedure.

Whenever a logic error is encountered, a message form similar to: ENTER (1, 500) AT PATH 100 TRYING TO ENTER BEYOND THE END OF THE PROCEDURE is printed, which includes the name of the operator, values of N1 and N2, the location in the procedure and the nature of the error. In diagnostic mode, errors are either ignored or corrective action is taken when possible. In execution mode, an error causes an additional printout of the ANAL68/360 status, analogue to the output from a STATUS operator.

In diagnostic mode, print from STATUS and COUNTS operators is allowed to facilitate the interpretation of diagnostic messages.

7. The Master Table - STABLE

7.1 Introduction

ANAL68/360 is a table driven command language. All requests to it are checked against a master table, STABLE, which contains one entry of twenty-four bytes, six words, for each available operator in the system. A program, TABLE, was written and is available to generate this table in suitable format for use by other subprograms in ANAL68/360.

7.2 Input

The input to TABLE consists of punched cards providing the necessary information to identify each available operator and specify the options and parameters required for its proper execution.

7.3 Output

The output from TABLE is a source deck which is a self-contained program to be loaded as part of ANAL68/360. This new program is the table that contains the information provided in the input cards, each operator being defined by a

twenty-four byte entry of packed binary information.

Execution time is about two minutes, including output time. The table contains 128 operators including ILEGAL and EXECUT, (entries 0 and 128 respectively) requiring 3072 bytes of core storage. Appendix III gives the format of the input and output cards for the operators available at the present time.

8. Checkout and Debugging of ANAL68/360

Checkout and debugging of ANAL68/360 programs was performed in three different computers under three operating systems. The bulk of the testing was done on the IBM360/67 time-sharing mode. This allowed for complete online debugging and provided for excellent response time. The time-sharing system allows infinite storage (theoretically), therefore no minimum core requirements need be set. The input for the test programs was pre-stored on disk and output was both online and on the system printer.

In parallel with the above efforts, programs were submitted to the IBM 360/65 under MFT release 14, and later to an IBM 360/91 under LASP/MVT, release 15/16.

The purpose of this dual checkout is:

1. To test program linkages and obtain an idea of the core requirements on non-time-shared systems.

2. Since the Computation Center at M. I. T. has an IBM 360/65 directly coupled via a channel to channel adapter to a 360/40 using ASP (Attached Support Processor) and this system is similar to the IBM 360/91 running under LASP, a modified version of ASP for a single processor, then the programs would be properly debugged when made available to other M. I. T. users.

At present the diagnostic phase of ANAL68/360 is operational and provides equivalent results as the original version on the UNIVAC 1108. Final checkout for the execution phase is being completed. This work is being done on the IBM 360/91 and 360/44 to provide other users with a choice of programs according to the size of their computers. The 360/44 version will include program overlaying while the model 91 version will take advantage of its size and speed. It should be mentioned that implementation of the execution phase for ANAL68/360 requires that the diagnostic phase be expanded, to include the input-output capabilities in order to handle the number of arrays the user would like to save, and to

add the subprograms which utilize these new capabilities. These programs are in FORTRAN and have already been written.

9. Future Development

It is expected that ANAL68/360 will be used to provide students and researchers in the fields of meteorology and oceanography with a tool in the solution of new problems. To this effect a complete reference manual will be made available with a more complete description of all 128 operators available, and a wide variety of examples.

Once this is accomplished it is planned to investigate the possibilities of utilizing smaller computers, such as the IBM 1130, 1800, and 360/20, to perform the diagnostic phase of ANAL68/360.

Direct communication between these smaller machines and larger IBM 360's will also be considered. This will allow for running of the diagnostic phase on a small system and the transmission of the checked out procedure to a faster machine for execution with the results being made available at the site chosen by the scientist. The software to accomplish this is partially available (12). What is needed is time to modify and

adapt it to the requirements of ANAL68/360. It should also include free formatted input. Software development should provide for the testing of new operators which will be more complex in nature, replacing not one, but several of the existing ones. As an example, the LOOP and END operators could be made a single operator with several parameters indicating the range of the loop and as many other options as required. The application of ANAL68/360 to other disciplines such as mathematics or physics should be encouraged.

10. Conclusions

The existence of an homogeneous data base and the international effort being made in the accumulation and storage of data necessitate the means by which it is to be processed.

ANAL68/360 is a means by which a meteorologist is able to avail himself of all the capabilities afforded by a computer. Moreover, he can do this in his own language and can then tackle problems of increasing complexity without the need to depend upon another person to interpret his equations into executable statements. The steps for building his own programs are clearly defined and the sue with which procedures can be built and modi-

fied permit him a new freedom in the approach towards the solution of unknowns. By simply using ANAL68/360 he can manipulate his variables with a satisfactory solution available. Programs such as ANAL68/360 should help in the development of meteorology as an exact science.

TABLE OF FIGURES

ANAL68/360 STATEMENT FORMAT.

ANAL68/360 programs are constructed based upon fixed format statements that define the desired processing. Each statement is identified by the contents of columns 1-6. Other information required by a function is obtained from other fields in the input statement. as shown in the following layout:

COLUMNS	DESCRIPTION
1-6	Statement identifier
7-24	Commentary, or alphanumeric arguments for function requiring them
25-30	For functions allowing an option equal to false, any non-blank character in this field will cause the option to be equal to true
31-34, 35-38	Two integers, N1 and N2, which usually specify an array, data block or logical unit number
39-45, 46-52, 53-59, 60-66, 67-73, 74-80	As many as six floating point arguments

Notes:

1. In the function description with names longer than six characters, only the first six are significant. With fewer than six, blanks are padded to the right.
2. The FORTRAN format for statements is (5A6,2F4.0, 6E7.0)
3. Allowable range for N1,N2 is less than 512. N1 and N2 can be modified by a counter by adding 1000x(counter number) to the value of N1 or N2. The result of this modification is always taken as positive (the sign is ignored). Thus if COUNT2 equals -2.0 and N1 is entered as 2003, N1 will be replaced, for the specific function where it is requested, with 3. - 2. = 1.
4. Each statement can contain 24 columns of alphanumeric arguments or six floating point arguments. For functions requiring more than

Figure 1.1

these limits, subsequent statements with ETCbbb in columns 1-6 must be present to provide the required number of arguments.

5. An argument like 2.6 can be pinched as 26-1 (equivalent to 26×10^{-1}).

Figure 1.2

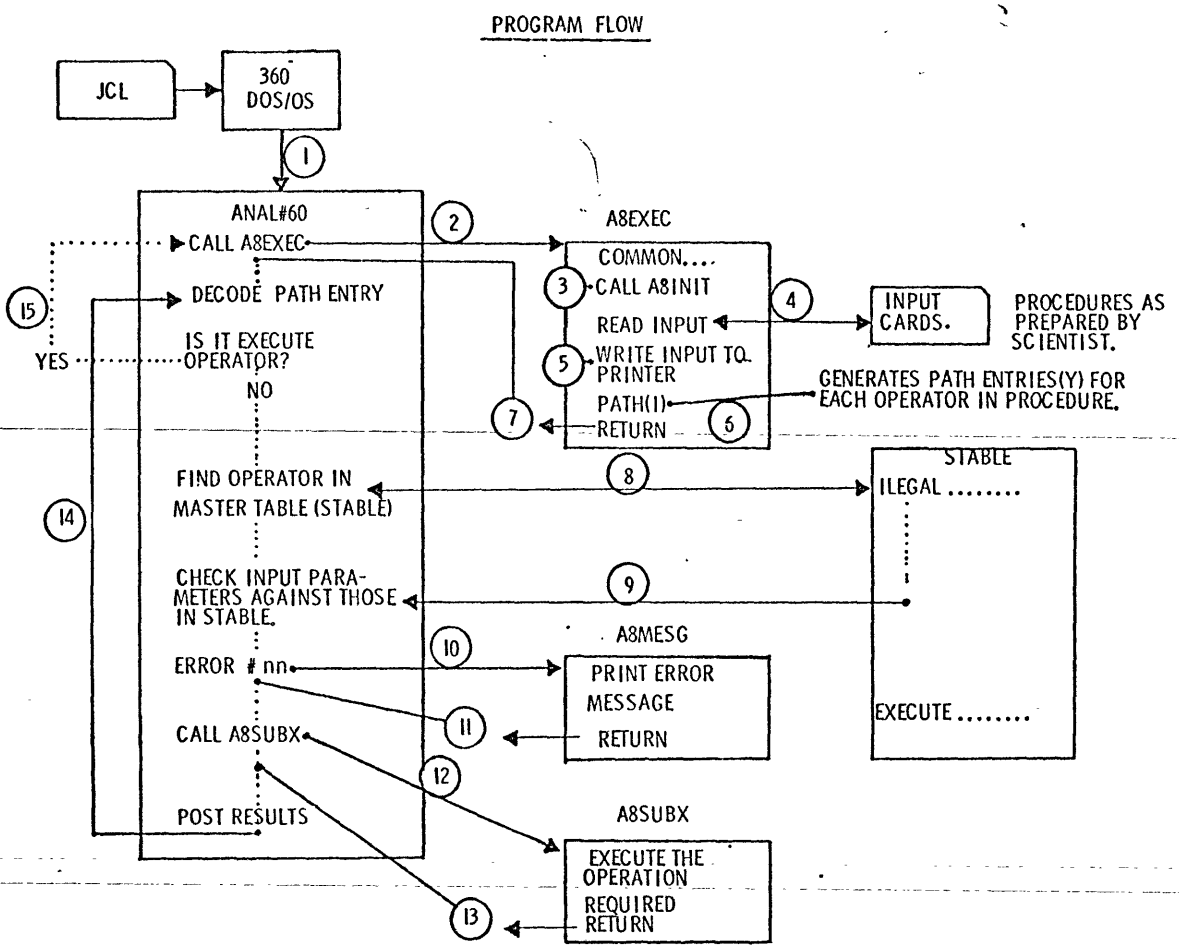

PROGRAM FLOW
JCL
360 DOS/OS
1
ANAL#60
CALL A8EXEC
DECODE PATH ENTRY
IS IT EXECUTE OPERATOR?
YES
NO
FIND OPERATOR IN MASTER TABLE (STABLE)
CHECK INPUT PARA-METERS AGAINST THOSE IN STABLE.
ERROR # nn
CALL A8SUBX
POST RESULTS
2
A8EXEC
COMMON....
3
CALL A8INIT
READ INPUT
4
INPUT CARDS.
PROCEDURES AS PREPARED BY SCIENTIST.
5
WRITE INPUT TO PRINTER
PATH(I)
GENERATES PATH ENTRIES(Y) FOR EACH OPERATOR IN PROCEDURE.
RETURN
6
7
8
STABLE
ILEGAL .....
EXECUTE .....
9
10
A8MESG
PRINT ERROR MESSAGE
RETURN
11
12
A8SUBX
EXECUTE THE OPERATION
REQUIRED RETURN
13
14
15

Figure 2

ANAL68 CODING FORM

AUTHOR	PROCEDURE NAME	DATE	PAGE OF

STMT	ALPHABETIC ARGUMENTS	OPTION	N1	N2	FLOATING-POINT ARGUMENTS

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

Figure 3

ANAL68 EXECUTIVE READING THE NEXT PROCEDURE

STEP	PATH	CTRS	STATEMENTS									
1	1	0	DIAGNOSIS		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	2	0	TITLE FOR NORM MACROENALOSTARR		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	3	0	LOOP FOR SEASONS		1.	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	4	1	ATABLE FALLSUMMERSPRINGWINTER		5.	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			ETC 6CMONS		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	10	1	VALUE		18.1000.		0.0	0.0	0.0	0.0	0.0	0.0
6	11	1	GRID	NODATA	0.0	0.0	0.0	50.0000E-02	0.0	90.0000E 00	0.0	0.0
			ETC		0.0	0.0	2.1000E 01	4.5000E 01	0.0	0.0	0.0	0.0
7	20	1	SET		0.0	11.	-40.0000E-01	40.0000E-01	0.0	0.0	0.0	0.0
8	23	1	FETCH U		2.2041.		0.0	0.0	0.0	0.0	0.0	0.0
9	24	1	FETCH V	NOWAIT	3.2042.		0.0	0.0	0.0	0.0	0.0	0.0
10	25	1	MOVE		2.	4.	0.0	0.0	0.0	0.0	0.0	0.0
11	26	1	MOVE		2.	5.	0.0	0.0	0.0	0.0	0.0	0.0
12	27	1	CHECK		0.02042.		0.0	0.0	0.0	0.0	0.0	0.0
13	29	1	MOVE		3.	6.	0.0	0.0	0.0	0.0	0.0	0.0
14	30	1	MULTIPLY		2.	4.	0.0	0.0	0.0	0.0	0.0	0.0
15	31	1	MULTIPLY		3.	5.	0.0	0.0	0.0	0.0	0.0	0.0
16	32	1	MULTIPLY		3.	6.	0.0	0.0	0.0	0.0	0.0	0.0
17	33	1	LOOP		1.	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	34	2	ZERO		2000.	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	35	2	SUMBI		2001.2000.		0.0	0.0	0.0	0.0	0.0	0.0
20	36	2	END		5.	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	37	1	GRID		0.0	0.0	0.0	50.0000E-02	0.0	90.0000E 00	10.0000E-01	0.0
			ETC		0.0	0.0	2.0000E 00	4.5000E 01	0.0	0.0	0.0	0.0
22	46	1	LOOP		5.	0.0	0.0	0.0	0.0	0.0	0.0	0.0
23	47	2	MOVE		2000.2001.		0.0	0.0	0.0	0.0	0.0	0.0
24	48	2	TXFORM		2001.	0.0	0.0	-50.0000E-03	0.0	0.0	0.0	0.0
25	51	2	TEST		0.0	0.0	10.0000E-01	-10.0000E-01	0.0	0.0	0.0	0.0
26	54	1	MOVE		2.	1.	0.0	0.0	0.0	0.0	0.0	0.0
27	55	1	MULTIPLY		3.	1.	0.0	0.0	0.0	0.0	0.0	0.0
28	56	1	MULTIPLY		2.	2.	0.0	0.0	0.0	0.0	0.0	0.0
29	57	1	MULTIPLY		3.	3.	0.0	0.0	0.0	0.0	0.0	0.0
30	58	1	ADD	SUB	2.	4.	0.0	0.0	0.0	0.0	0.0	0.0
31	59	1	ADD	SUB	1.	5.	0.0	0.0	0.0	0.0	0.0	0.0
32	60	1	ADD	SUB	3.	6.	0.0	0.0	0.0	0.0	0.0	0.0
33	61	1	MULTIPLY		6.	4.	0.0	0.0	0.0	0.0	0.0	0.0
34	62	1	SQRT		4.	4.	0.0	0.0	0.0	0.0	0.0	0.0
35	63	1	MULTIPLY	DIVIDE	4.	5.	0.0	0.0	0.0	0.0	0.0	0.0
36	64	1	SHADE	NOPERF	5.	0.0	0.0	10.0000E 01	0.0	0.0	0.0	0.0
37	69	1	GRID		0.0	0.0	0.0	50.0000E-02	0.0	90.0000E 00	0.0	0.0
			ETC		0.0	0.0	2.1000E 01	4.5000E 01	0.0	0.0	0.0	0.0
38	78	1	SET		0.0	11.	-40.0000E-01	40.0000E-01	0.0	0.0	0.0	0.0
39	81	1	FETCH U		1.2041.		0.0	0.0	0.0	0.0	0.0	0.0
40	82	1	FETCH V		2.2042.		0.0	0.0	0.0	0.0	0.0	0.0
41	83	1	ZERO		5.	6.	0.0	0.0	0.0	0.0	0.0	0.0
42	84	1	SUMBI		1.	5.	0.0	0.0	0.0	0.0	0.0	0.0
43	85	1	SUMBI		2.	6.	0.0	0.0	0.0	0.0	0.0	0.0
44	86	1	TXFORM		5.	0.0	0.0	-50.0000E-03	0.0	0.0	0.0	0.0
45	89	1	TXFORM		6.	0.0	0.0	-50.0000E-03	0.0	0.0	0.0	0.0
46	92	1	FETCH SINCOS	NOWAIT	3.	5.	0.0	0.0	0.0	0.0	0.0	0.0
47	93	1	ZERO		5.	0.0	0.0	0.0	0.0	0.0	0.0	0.0
48	94	1	CI	ZERO	6.	4.	0.0	0.0	0.0	0.0	0.0	0.0
49	95	1	ADD		4.	6.	0.0	0.0	0.0	0.0	0.0	0.0
50	96	1	CBI	ZERO	6.	4.	0.0	0.0	0.0	0.0	0.0	0.0

Figure 4.1

Figure 4.2

```
ANALYZE EXECUTIVE READING THE NEXT PROCEDURE

STEP PATH CTRS   S T A T E M E N T S
  51    97  1  SUMI                              4.   5.  0.0        0.0        0.0        0.0        0.0        0.0
  52    98  1  ADD                        SUB    5.   6.  0.0        0.0        0.0        0.0        0.0        0.0
  53    99  1  SWAP                              1.   2.  0.0        0.0        0.0        0.0        0.0        0.0
  54   100  1  CHECK                             0.0  5.  0.0        0.0        0.0        0.0        0.0        0.0
  55   102  1  MULTIPLY                          3.   2.  0.0        0.0        0.0        0.0        0.0        0.0
  56   103  1  MULTIPLY                          2.   1.  0.0        0.0        0.0        0.0        0.0        0.0
  57   104  1  MULTIPLY                          6.   2.  0.0        0.0        0.0        0.0        0.0        0.0
  58   105  1  MOVE                              1.   3.  0.0        0.0        0.0        0.0        0.0        0.0
  59   106  1  ADD                        SUB    2.   3.  0.0        0.0        0.0        0.0        0.0        0.0
  60   107  1  LOOP                              1.   0.0 0.0        0.0        0.0        0.0        0.0        0.0
  61   108  2  ATABLEINTGNORGTSIDLFTSID           3.   0.0 0.0        0.0        0.0        0.0        0.0        0.0
  62   112  2  VALUE                             19.2000. 0.0        0.0        0.0        0.0        0.0        0.0
  63   113  2  ENTER                             19.   2.  0.0        0.0        0.0        0.0        0.0        0.0
  64   114  2  NAME  CLOGRD                  2027.  0.0  0.0        0.0        0.0        0.0        0.0        0.0
  65   116  2  SPACE                      NOPERF2000.  0.0 0.0        10.0000E-01-99.0000E 01 50.0000E-01   0.0        0.0
  66   121  2  END                               3.   0.0 0.0        0.0        0.0        0.0        0.0        0.0
  67   122  1  AVERAGE                           1.   0.0 0.0        0.0        0.0        0.0        0.0        0.0
              ETC                               0.0  0.0 1.0000E 00 0.0        2.0000E 01 4.4000E 01 0.0        0.0
  68   133  1  AVERAGE                           2.   0.0 0.0        0.0        0.0        0.0        0.0        0.0
              ETC                               0.0  0.0 1.0000E 00 0.0        2.0000E 01 4.4000E 01 0.0        0.0
  69   144  1  AVERAGE                           3.   0.0 0.0        0.0        0.0        0.0        0.0        0.0
              ETC                               0.0  0.0 1.0000E 00 0.0        2.0000E 01 4.4000E 01 0.0        0.0
  70   155  1  COUNTS                            0.0  0.0 0.0        0.0        0.0        0.0        0.0        0.0
  71   156  1  SUMMARY                           0.0  0.0 0.0        0.0        0.0        0.0        0.0        0.0
  72   157  1  END                               1.   0.0 0.0        0.0        0.0        0.0        0.0        0.0
  73   158  0  STATUS                     DRUMS  0.0  0.0 0.0        0.0        0.0        0.0        0.0        0.0
  74   159  0  EXECUTE                           0.0  0.0 0.0        0.0        0.0        0.0        0.0        0.0
```

PATH ENTRIES FOR ABOVE PROCEDURE

```
d path:path.(x'fff')
PATH:PATH.(X'FFF')          4096 BYTES LOCATIONS ARE RELATIVE TO A8CMIS   AT 000DF000
01 006A0  11000000 00000000 63000000 00000000 33001000 00000000 04005000 00000000 4040C6C1 D3D34040 E2E4D4D4 C5D94040
01 006D0  E2D7D9C9 D5C74040 E6C9D5E3 C5D94040 F6F0D4D6 D5E24040 6B012200 00000000 DB000000 00000000 43000000 00000000
01 00700  40800000 00000000 40000000 00000000 425A0000 00000000 42000000 00000000 42000000 00000000 42150000 00000000
01 00730  422D0000 00000000 5000000B 00000000 C1400000 00000000 41400000 00000000 21002429 00000000 DFFFCBD6 00000000
01 00760  35002004 00000000 35002005 00000000 0A00042A 00000000 43000000 00000000 35003006 00000000 36002004 00000000
01 00790  36003005 00000000 36003006 00000000 33001000 00000000 72400000 00000000 5D401400 00000000 1B005021 00000000
01 007C0  25000000 00000000 41000000 00000000 40800000 00000000 40000000 00000000 425A0000 00000000 41100000 00000000
01 007F0  41000000 00000000 41200000 00000000 422D0000 00000000 33005000 00000000 35400401 00000000 67401000 00000000
01 00820  43000000 00000000 BFCCCCCC CCCCCCCC 6200002C 00000000 41100000 00000000 C1100000 00000000 35002001 00000000
01 00850  36003001 00000000 36002002 00000000 36003003 00000000 FEFEDFFC 00000000 FEFEFFFB 00000000 FFFFCFFA 00000000
01 00880  36006004 00000000 57004004 00000000 C9FFBFFB 00000000 AFFFB000 00000000 41000000 00000000 42640000 00000000
01 008B0  42000000 00000000 42000000 00000000 25000000 00000000 42000000 00000000 40800000 00000000 40000000 00000000
01 008E0  425A0000 00000000 42000000 00000000 42000000 00000000 42150000 00000000 422D0000 00000000 5000000B 00000000
01 00910  C1400000 00000000 41400000 00000000 21001429 00000000 2100242A 00000000 72005006 00000000 5D001005 00000000
01 00940  5D002006 00000000 67005000 00000000 41000000 00000000 BFCCCCCC CCCCCCCC 67006000 00000000 41000000 00000000
01 00970  BFCCCCCC CCCCCCCC DFFFCFFB 00000000 72005000 00000000 EAFF9FFC 00000000 01004006 00000000 E8FF9FFC 00000000
01 009A0  5B004005 00000000 FEFFAFFA 00000000 60001002 00000000 0A000005 00000000 41000000 00000000 36003002 00000000
01 009D0  36002001 00000000 36006002 00000000 35001003 00000000 FEFFDFFD 00000000 33001000 00000000 04003000 00000000
01 00A00  C9D5E3C7 D5C44040 D9C7E3E2 C9C44040 D3C6E3E2 C9C44040 6B013400 00000000 1D013002 00000000 3741B000 00000000
01 00A30  D6D3C4C7 D9C44040 AEC00000 00000000 43000000 00000000 41100000 00000000 C33DE000 00000000 41500000 00000000
01 00A60  1B00306B 00000000 03001000 00000000 41000000 00000000 41000000 00000000 41000000 00000000 41000000 00000000
01 00A90  41000000 00000000 41000000 00000000 41100000 00000000 41000000 00000000 42140000 00000000 422C0000 00000000
01 00AC0  03002000 00000000 41000000 00000000 41000000 00000000 41000000 00000000 41000000 00000000 41000000 00000000
01 00AF0  41000000 00000000 41100000 00000000 41000000 00000000 42140000 00000000 422C0000 00000000 03003000 00000000
01 00B20  41000000 00000000 41000000 00000000 41000000 00000000 41000000 00000000 41000000 00000000 41000000 00000000
01 00B50  41100000 00000000 41000000 00000000 42140000 00000000 422C0000 00000000 0E000000 00000000 5F000000 00000000
01 00B80  1B001003 00000000 A6000000 00000000 80000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01 00BB0:01 0169F CONTAINS 00000000
```

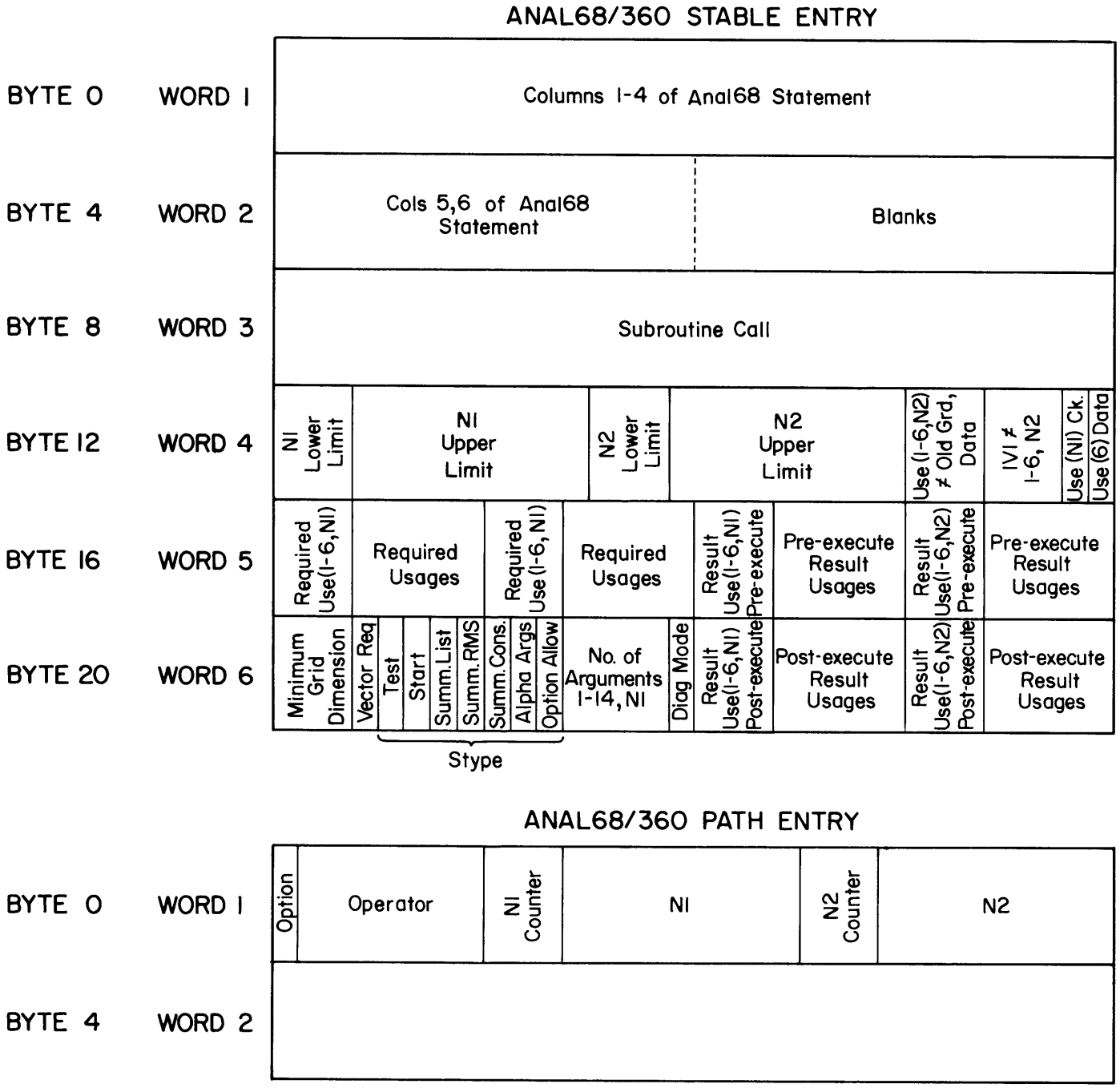

ANAL68/360 STABLE ENTRY
BYTE 0
WORD 1
Columns 1-4 of Anal68 Statement
BYTE 4
WORD 2
Cols 5,6 of Anal68 Statement
Blanks
BYTE 8
WORD 3
Subroutine Call
BYTE 12
WORD 4
NI Lower Limit
NI Upper Limit
N2 Lower Limit
N2 Upper Limit
Use (I-6,N2) ≠ Old Grd, Data
IVI ≠ I-6, N2
Use (NI) Ck.
Use (6) Data
BYTE 16
WORD 5
Required Use(I-6,NI)
Required Usages
Required Use (I-6, NI)
Required Usages
Result Use(I-6,NI) Pre-execute
Pre-execute Result Usages
Result Use(I-6,N2) Pre-execute
Pre-execute Result Usages
BYTE 20
WORD 6
Minimum Grid Dimension
Vector Req
Test
Start
Summ.List
Summ.RMS
Summ.Cons.
Alpha Args
Option Allow
Stype
No. of Arguments I-14, NI
Diag Mode
Result Use(I-6,NI) Post-execute
Post-execute Result Usages
Result Use(I-6,N2) Post-execute
Post-execute Result Usages
ANAL68/360 PATH ENTRY
BYTE 0
WORD 1
Option
Operator
NI Counter
NI
N2 Counter
N2
BYTE 4
WORD 2

Figure 5

APPENDIX I

DESCRIPTION OF VARIABLES IN NAMED COMMON - A8CMIS -

VARIABLE	USE
OPNAME	NAME OF OPERATOR BEING PROCESSED
TITLE	PORTION OF PAGE HEADER
CARD	INPUT BUFFER FOR CARDS READ IN
VALUES	STORAGE AREA FOR SELECTED PARAMETERS
COUNTR	NAME AND VALUES OF COUNTERS
USAGES	NAME OF AVAILABLE USES
USE	NAME OF REQUESTED USES
OUSE	NAME OF PREVIOUSLY REQUESTED USES (OLD)
BLANK	CONSTANT EQUAL TO BLANKS
N1	SPECIFIES COUNTER AND ARRAY REQUIRED
N2	SPECIFIES COUNTER AND ARRAY REQUIRED
IP	PATH ENTRY POINTER
NSTEP	INDICATES STEP IN PROCEDURE
MODE	OPERATING MODE
NCTRS	NUMBER OF ACTIVE COUNTERS
NPHI	NUMBER OF PHYSICAL VARIABLES
NPLOTS	NUMBER OF PLOTS REQUESTED
NPLINE	NUMBER OF LINES PER INCH ON PRINTED OUTPUT
NEXEC	LOCATION OF 'EXECUT' OPERATOR IN STABLE
NSUB	LOCATION OF OPERATOR NAMED BY OPNAME IN STABLE
MAXIL	MAXIMUM VALUE FOR I-AXIS OF GRID
MAXJL	MAXIMUM VALUE FOR J-AXIS OF GRID
MAXPTH	MAXIMUM NUMBER OF ENTRIES IN PATH TABLE
BAND	SHADE CONTROL CHARACTERS
IBLK	NUMBER OF STATIONS FOR OBSERVATIONAL DATA
ROTN	THE GRID Y-AXIS ROTATION WESTWARD FROM GREENWICH
AGRID	THE RATIO BETWEEN THE DESIRED GRID MESH AND THE JNWP GRID
XPOLE	THE X-COORDINATE OF THE NORTH POLE ON THE GRID WITH UNIT MESH
YPOLE	THE Y-COORDINATE OF THE NORTH POLE ON THE GRID WITH UNIT MESH
ILONG	PROGRAM GRID LENGTH IN THE X-DIRECTION
JLONG	PROGRAM GRID LENGTH IN THE Y-DIRECTION
XLEFT	X-COORDINATE OF THE UPPER LEFT CORNER OF PROGRAM GRID
YTOP	Y-COORDINATE OF THE UPPER LEFT CORNER OF PROGRAM GRID

VARIABLE	USE
OGRID	PARAMETERS DEFINING USER'S PROGRAM GRIDS UP TO A MAXIMUM OF 6
PLINE	WORK AREA
NCCLIB	NOT USED
DUM (IDUM)	NUMBER OF ERROR MESSAGE REQUESTED BY THE MAIN PROGRAM (ANAL360)
PATH	PATH ENTRY TABLE: AT LEAST ONE ENTRY PER REQUESTED OPERATOR: MAXIMUM LENGTH OF TABLE DETERMINED BY MAXPTH

APPENDIX II

APPENDIX II gives a description of the functions available under ANAL68/360 and, provides examples of the requirements set by some of the operators.

FORTRAN Functions Available in ANAL68/360

The following functions are available in ANAL68/360.

ANAL68/360 Statement	FORTRAN Function	
SIN	SIN	A(N2) = SIN(A(N1))
COS	COS	A(N2) = COS(A(N1))
TAN	TAN	A(N2) = TAN(A(N1))
ATAN	ATAN2	A(N2) = ATAN2(A(N1);A(N2))
EXP	EXP	A(N2) = EXP(A(N1))
LOG	ALOG	A(N2) = ALOG(A(N1))
SQRT	SQRT	A(N2) = SQRT(A(N1))

Notes:

1. Error messages and other diagnostics are provided by the FORTRAN subroutines and are outside the control of ANAL68/360.
2. SIN,COS and TAN require A(N1) to be in radians, and ATAN yields the arctangent of A(N1)/A(N2) also in radians.
3. EXP and LOG are natural exponents and logarithms.
4. The following are FORTRAN restrictions:

SIN,COS	$\|A(N1)\| < 2^{18}\pi$ ($< 2^{50}\pi$)
TAN	$\|A(N1)\| < 2^{18}\pi$ ($< 2^{50}\pi$)
ATAN	no restrictions
EXP	A(N1) < 174.673 (< 174.67309)
LOG	A(N1) > 0.
SQRT	A(N1) > 0.

The limits in parenthesis are for double precision variables. All others apply to single precision and or double precision.(1)

(1)IBM, 'IBM System/360 Time Sharing System, FORTRAN IV Library Subprograms.'.Time Sharing System Publications, Form C28-2026,Yorktown Heights, New York,1968.

OPERATOR NAME: LAPLACIAN

DESCRIPTION:

The Laplacian of A (N1) is computed and stored in A(3).

OPTION: NONE

N1 : 1 - 6

N2 : NONE

ARGUMENTS: NONE

REQUIRED FIELDS: A(N1)

GENERATED FIELDS: A(3)

INPUT-OUTPUT: NONE

ERRORS: Misspecified array.

COMMENTS:

A(3) will be "VRTCTY" if A(N1) is "STREAM" , A(3) will be "DVRGNC" if A(N1) is "POTNTL" , otherwise A(3) will be "LAPLAC"

OPERATOR NAME: U,V

DESCRIPTION: The vector (A(1),A(2)) is transformed to components if it is in the form direction, speed.

OPTION: NONE

N1 : NONE

N2 : NONE

ARGUMENTS: NONE

REQUIRED FIELDS: A(1) must be "DIRECN" or "DIRECO" and A(2) must be "MAGNTD".

GENERATED FIELDS: If A(1) was "DIRECN", A(1) will be "U", A(2) "V" and if A(1) was "DIRECO", A(1) will be "UO", A(2) "VO".

INPUT-OUTPUT: NONE

ERRORS: Unavailable arrays.

COMMENTS: If A(1) and A(2) are already "U" and "V" or "UO" and "VO", no action is taken.

OPERATOR NAME: END

DESCRIPTION:

The currently activated counter is incremented by 1. If the value of the counter exceeds the value of N1, the procedure continues with the next statement. If the value of the counter does not exceed N1, processing is transferred to the first statement after the START or LOOP with which the END statement is matched.

OPTION: NONE

N1 : 0 - 511

N2 : Set by ANAL68/360 bookkeeping.

ARGUMENTS: NONE

REQUIRED FIELDS: NONE

GENERATED FIELDS: NONE

INPUT-OUTPUT: NONE

ERRORS: NONE

COMMENTS:

END is similar to and interchangeable with TEST, except that the counter increment is always 1 and the maximum

counter value must be some integer between 0 and 511. N1 modification is permitted, so that the maximum counter value can be the value of some other counter.

OPERATOR NAME: EXECUTE

DESCRIPTION:

Control cards are read and interpreted to set up the PATH array until:

a. an EXECUTE card is encountered

b. an EXIT card is encountered

or, c. the maximum number of PATH entries are made.

When one of the three conditions is met, the PATH is executed unless the counters are unbalanced.

OPTION: NONE

N1 : NONE

N2 : NONE

ARGUMENTS: NONE

REQUIRED FIELDS: NONE

GENERATED FIELDS:NONE

INPUT-OUTPUT: NONE

ERRORS: 4 - PATH too long or untested counters
5 - more than 7 counters activated
6 - no ETC card where expected
1 - unrecognizable control card

COMMENTS: NONE

APPENDIX III

TABLE 1. LAYOUT OF INPUT TO STABLE

COLUMNS	DESCRIPTION OF FIELD
1	Not used
2-7	Operator name
8-9	Not used
10-15	Subroutine name
16	Not used
17	N1 lower limit
18	Not used
19-21	N1 upper limit
22	Not used
23	N2 lower limit
24	Not used
25-27	N2 upper limit
28	Not used
29	USE(1-6,N2) NOT OLDGRD,DATA
30	Not used
31	N1 NOT (1-6,N2)
32	Not used
33	USE(N1);USE(6),DATA;Vector flags
34	Not used
35	Minimum grid dimensions
36	Not used
37	Required Use(1-6,N1)
38	Not used
39-40	Required Usages(1-6,N1)
41	Not used
42	Required Use(1-6,N2)
43	Not used
44-45	Required Usages(1-6,N2)
46	Not used
47	Pre-execute result Use(1-6,N1)
48	Not used
49-50	Pre-execute result Usages(1-6,N1)
51	Not used
52	Pre-execute result Use(1-6,N2)
53	Not used
54-55	Pre-execute result Usages(1-6,N2)
56	Not used
57-59	STYPE - 7 flag bytes
60	Not used
61-62	Number of arguments
63	Not used
64	Diagnostic mode flag
65	Not used

COLUMNS	DESCRIPTION OF FIELD
66	Post-execute result Use(1-6,N1)
67	Not used
68-69	Post-execute result Usages(1-6,N1)
70	Not used
71	Post-execute result Use(1-6,N2)
72	Not used
73-76	Post-execute result Usages(1-6,N2)
77-80	Not used

Note:

The layout of the input cards to the program that generates STABLE follow the sequence in which the parameters are packed in the master table with the exception of the vector flag and the grid dimension. Refer to Figure 5 for the layout of entries in STABLE.

```
*****************************************************************
*                                                               *
*                                                               *
*            TABLE 2. TABLE OF OPERATORS                        *
*                                                               *
*                          - STABLE -                           *
*                                                               *
*                                                               *
*****************************************************************


STABLE   CSECT
         ENTRY ENDSTB
         DC    CL8'ILEGAL   '
         DC    V(A8B00B)
         DC    XL4'00000000'
         DC    XL4'00000000'
         DC    XL4'00000000'
         DC    CL8'ADD      '
         DC    V(A8B00B)
         DC    XL4'206206E2'
         DC    XL4'00000000'
         DC    XL4'202000E0'
         DC    CL8'ALTER    '
         DC    V(A8ALTR)
         DC    XL4'21101100'
         DC    XL4'00000000'
         DC    XL4'00050000'
         DC    CL8'AVERAG   '
         DC    V(A8B00B)
         DC    XL4'20600602'
         DC    XL4'00E80000'
         DC    XL4'21140000'
         DC    CL8'ATABLE   '
         DC    V(A8RETN)
         DC    XL4'3FF00000'
         DC    XL4'00000000'
         DC    XL4'005E0000'
         DC    CL8'ATAN     '
         DC    V(A8B00B)
         DC    XL4'206206FE'
         DC    XL4'000000E0'
         DC    XL4'200000E0'
         DC    CL8'AUTOCD   '
         DC    V(A8B00B)
         DC    XL4'22822803'
         DC    XL4'A4000080'
         DC    XL4'E0208000'
         DC    CL8'AUTOCI   '
         DC    V(A8B00B)
         DC    XL4'20620602'
         DC    XL4'000000E0'
```

```
DC      XL4'200000E0'
DC      CL8'AUTOCJ  '
DC      V(A8B00B)
DC      XL4'20620602'
DC      XL4'000000E0'
DC      XL4'200000E0'
DC      CL8'BANDS   '
DC      V(A8B00B)
DC      XL4'00000000'
DC      XL4'00000000'
DC      XL4'00480000'
DC      CL8'CHECK   '
DC      V(A8B00B)
DC      XL4'00027800'
DC      XL4'00000000'
DC      XL4'00820000'
DC      CL8'COPY    '
DC      V(A8B00B)
DC      XL4'0820821C'
DC      XL4'00000000'
DC      XL4'00020000'
DC      CL8'CORREC  '
DC      V(A8B00B)
DC      XL4'22822881'
DC      XL4'A4000000'
DC      XL4'40222343'
DC      CL8'COS     '
DC      V(A8B00B)
DC      XL4'20620602'
DC      XL4'000000E0'
DC      XL4'200000E0'
DC      CL8'COUNTS  '
DC      V(A8CNTS)
DC      XL4'00000000'
DC      XL4'00000000'
DC      XL4'00210000'
DC      CL8'DATA    '
DC      V(A8B00B)
DC      XL4'22808200'
DC      XL4'00000000'
DC      XL4'0020A4C0'
DC      CL8'DD/FF   '
DC      V(A8DRSP)
DC      XL4'00000000'
DC      XL4'00000000'
DC      XL4'30012040'
DC      CL8'DIAGNO  '
DC      V(A8DIAG)
DC      XL4'00000000'
DC      XL4'00000000'
DC      XL4'00000000'
DC      CL8'DIFFER  '
DC      V(A8B00B)
DC      XL4'22822881'
```

```
DC      XL4'A4000000'
DC      XL4'41340000'
DC      CL8'DISCAR  '
DC      V(A8B00B)
DC      XL4'22822881'
DC      XL4'A4000000'
DC      XL4'402A0000'
DC      CL8'DIVERG  '
DC      V(A8B00B)
DC      XL4'00000000'
DC      XL4'2E4F0000'
DC      XL4'60006900'
DC      CL8'DI      '
DC      V(A8B00B)
DC      XL4'20620602'
DC      XL4'000000E0'
DC      XL4'202000E0'
DC      CL8'DJ      '
DC      V(A8B00B)
DC      XL4'20620602'
DC      XL4'000000E0'
DC      XL4'202000E0'
DC      CL8'DBI     '
DC      V(A8B00B)
DC      XL4'20620602'
DC      XL4'000000E0'
DC      XL4'202000E0'
DC      CL8'DBJ     '
DC      V(A8B00B)
DC      XL4'20620602'
DC      XL4'000000E0'
DC      XL4'202000E0'
DC      CL8'DOCUME  '
DC      V(A8B00B)
DC      XL4'00A00A00'
DC      XL4'00000020'
DC      XL4'00200020'
DC      CL8'EDIT    '
DC      V(A8B00B)
DC      XL4'22800001'
DC      XL4'A4000000'
DC      XL4'00200000'
DC      CL8'END     '
DC      V(A8ELOP)
DC      XL4'1FF1FF00'
DC      XL4'00000000'
DC      XL4'08010000'
DC      CL8'END FI  '
DC      V(A8B00B)
DC      XL4'00008200'
DC      XL4'00000000'
DC      XL4'00200000'
DC      CL8'ENTER   '
DC      V(A8ENTR)
```

```
DC      XL4'2385FF00'
DC      XL4'00000000'
DC      XL4'00010000'
DC      CL8'EXP     '
DC      V(A8B00B)
DC      XL4'20620602'
DC      XL4'000000E0'
DC      XL4'200000E0'
DC      CL8'EXTREM  '
DC      V(A8B00B)
DC      XL4'20422802'
DC      XL4'A4000000'
DC      XL4'6000C000'
DC      CL8'FDCORR  '
DC      V(A8B00B)
DC      XL4'00008280'
DC      XL4'00000000'
DC      XL4'40202343'
DC      CL8'FETCH   '
DC      V(A8FECH)
DC      XL4'20627800'
DC      XL4'0000E000'
DC      XL4'0021E000'
DC      CL8'FIELD   '
DC      V(A8B00B)
DC      XL4'20608200'
DC      XL4'0000E000'
DC      XL4'0020E000'
DC      CL8'FTABLE  '
DC      V(A8RETN)
DC      XL4'3FF00000'
DC      XL4'00000000'
DC      XL4'001E0000'
DC      CL8'GRADIE  '
DC      V(A8B00B)
DC      XL4'60600002'
DC      XL4'00000000'
DC      XL4'60202E4F'
DC      CL8'GRID    '
DC      V(A8GRID)
DC      XL4'00000000'
DC      XL4'00000000'
DC      XL4'003100A0'
DC      CL8'INTERP  '
DC      V(A8B00B)
DC      XL4'20622803'
DC      XL4'A4000000'
DC      XL4'40000000'
DC      CL8'JACOBA  '
DC      V(A8B00B)
DC      XL4'206206FE'
DC      XL4'00006000'
DC      XL4'60006000'
DC      CL8'JACOBX  '
```

```
DC      V(A8B00B)
DC      XL4'206206FE'
DC      XL4'00006000'
DC      XL4'60006000'
DC      CL8'JACOB1  '
DC      V(A8B00B)
DC      XL4'206206FE'
DC      XL4'00006000'
DC      XL4'60006000'
DC      CL8'KINETI  '
DC      V(A8B00B)
DC      XL4'20620602'
DC      XL4'000000E0'
DC      XL4'600000E0'
DC      CL8'-KX     '
DC      V(A8B00B)
DC      XL4'00000000'
DC      XL4'00000000'
DC      XL4'30002040'
DC      CL8'LABEL   '
DC      V(A8B00B)
DC      XL4'00008200'
DC      XL4'00000000'
DC      XL4'00480000'
DC      CL8'LAPLAC  '
DC      V(A8LAPL)
DC      XL4'20600002'
DC      XL4'00000000'
DC      XL4'60216000'
DC      CL8'LATLON  '
DC      V(A8B00B)
DC      XL4'2062061C'
DC      XL4'0000E0E0'
DC      XL4'2000E0E0'
DC      CL8'LIMIT   '
DC      V(A8B00B)
DC      XL4'20600002'
DC      XL4'00000000'
DC      XL4'2004E000'
DC      CL8'LIST    '
DC      V(A8B00B)
DC      XL4'00000001'
DC      XL4'A4000000'
DC      XL4'00000000'
DC      CL8'LOG     '
DC      V(A8B00B)
DC      XL4'20620602'
DC      XL4'000000E0'
DC      XL4'200000E0'
DC      CL8'LOOKUP  '
DC      V(A8LKUP)
DC      XL4'205005E2'
DC      XL4'00000000'
DC      XL4'2029E000'
```

```
DC      CL8'LOOP    '
DC      V(A8LOOP)
DC      XL4'1FF00000'
DC      XL4'00000000'
DC      XL4'04010000'
DC      CL8'MAPFAC  '
DC      V(A8B00B)
DC      XL4'206006E2'
DC      XL4'00000000'
DC      XL4'2020E0E0'
DC      CL8'MOVE    '
DC      V(A8INTP)
DC      XL4'2062061C'
DC      XL4'000000E0'
DC      XL4'202100E0'
DC      CL8'MULTIP  '
DC      V(A8B00B)
DC      XL4'206206E2'
DC      XL4'00000000'
DC      XL4'202000E0'
DC      CL8'NAME    '
DC      V(A8NAME)
DC      XL4'22700000'
DC      XL4'00000000'
DC      XL4'004300E0'
DC      CL8'NEW PA  '
DC      V(A8B00B)
DC      XL4'00000000'
DC      XL4'00000000'
DC      XL4'00000000'
DC      CL8'NOEQT   '
DC      V(A8EXEC)
DC      XL4'00000000'
DC      XL4'00000000'
DC      XL4'00000000'
DC      CL8'OVERAL  '
DC      V(A8B00B)
DC      XL4'0331FF80'
DC      XL4'23430040'
DC      XL4'60202840'
DC      CL8'PEF     '
DC      V(A8B00B)
DC      XL4'20900A00'
DC      XL4'00000000'
DC      XL4'00000000'
DC      CL8'PHI     '
DC      V(A8PHIX)
DC      XL4'20400000'
DC      XL4'A4000000'
DC      XL4'00638000'
DC      CL8'PHIDIV  '
DC      V(A8B00B)
DC      XL4'20420400'
DC      XL4'A4C50000'
```

```
DC      XL4'00200000'
DC      CL8'PHISUB  '
DC      V(A8B00B)
DC      XL4'20420400'
DC      XL4'A4C50000'
DC      XL4'00200000'
DC      CL8'PHISWA  '
DC      V(A8B00B)
DC      XL4'2042041C'
DC      XL4'A4C50000'
DC      XL4'00000000'
DC      CL8'PHITXF  '
DC      V(A8B00B)
DC      XL4'20420400'
DC      XL4'A4C50000'
DC      XL4'00040000'
DC      CL8'PHIU,V  '
DC      V(A8B00B)
DC      XL4'2042041C'
DC      XL4'A4C50000'
DC      XL4'00000000'
DC      CL8'PLANE   '
DC      V(A8B00B)
DC      XL4'20600000'
DC      XL4'0000E000'
DC      XL4'4006E000'
DC      CL8'PLINES  '
DC      V(A8B00B)
DC      XL4'C1400000'
DC      XL4'00000000'
DC      XL4'00200000'
DC      CL8'POISSO  '
DC      V(A8POIS)
DC      XL4'2060060E'
DC      XL4'00E80000'
DC      XL4'60ADE000'
DC      CL8'PRINT   '
DC      V(A8B00B)
DC      XL4'206006E2'
DC      XL4'00000000'
DC      XL4'20080000'
DC      CL8'PROCED  '
DC      V(A8B00B)
DC      XL4'00000000'
DC      XL4'00000000'
DC      XL4'00210000'
DC      CL8'P.SMOO  '
DC      V(A8B00B)
DC      XL4'00100100'
DC      XL4'00000000'
DC      XL4'00240000'
DC      CL8'PTAGS   '
DC      V(A8B00B)
DC      XL4'20620602'
```

```
DC      XL4'00000000'
DC      XL4'200400E8'
DC      CL8'PUNCH    '
DC      V(A8B00B)
DC      XL4'20608202'
DC      XL4'00000000'
DC      XL4'20200000'
DC      CL8'PUNCH3   '
DC      V(A8B00B)
DC      XL4'20608202'
DC      XL4'00000000'
DC      XL4'20200000'
DC      CL8'Q-DATA   '
DC      V(A8B00B)
DC      XL4'1FF08200'
DC      XL4'24000000'
DC      XL4'0020A4C5'
DC      CL8'READ     '
DC      V(A8RDTA)
DC      XL4'00608200'
DC      XL4'0000E000'
DC      XL4'0001E000'
DC      CL8'REWIND   '
DC      V(A8B00B)
DC      XL4'00008200'
DC      XL4'00000000'
DC      XL4'00000000'
DC      CL8'RSMOOT   '
DC      V(A8B00B)
DC      XL4'2060061E'
DC      XL4'00E80000'
DC      XL4'6023E000'
DC      CL8'SAVE     '
DC      V(A8SAVE)
DC      XL4'20627800'
DC      XL4'00000000'
DC      XL4'00210000'
DC      CL8'SET      '
DC      V(A8SETX)
DC      XL4'01101100'
DC      XL4'00000000'
DC      XL4'00050000'
DC      CL8'SHADE    '
DC      V(A8B00B)
DC      XL4'20600002'
DC      XL4'00000000'
DC      XL4'40280000'
DC      CL8'SIDELA   '
DC      V(A8B00B)
DC      XL4'2060006E'
DC      XL4'00000000'
DC      XL4'60006000'
DC      CL8'SIN      '
DC      V(A8B00B)
```

```
         DC    XL4'20620602'
         DC    XL4'000000E0'
         DC    XL4'200000E0'
         DC    CL8'SINLAT  '
         DC    V(A8B00B)
         DC    XL4'20600000'
         DC    XL4'0000E000'
         DC    XL4'2000E000'
         DC    CL8'SKIPRE  '
         DC    V(A8B00B)
         DC    XL4'3FF08200'
         DC    XL4'00000000'
         DC    XL4'00200000'
         DC    CL8'SMOOTH  '
         DC    V(A8B00B)
         DC    XL4'3FF206E0'
         DC    XL4'00000000'
         DC    XL4'603F00E0'
         DC    CL8'SQRT    '
         DC    V(A8B00B)
         DC    XL4'20620602'
         DC    XL4'000000E0'
         DC    XL4'200000E0'
         DC    CL8'STABLE  '
         DC    V(A8B00B)
         DC    XL4'00000000'
         DC    XL4'00000000'
         DC    XL4'00000000'
         DC    CL8'START   '
         DC    V(A8STCT)
         DC    XL4'00000000'
         DC    XL4'00000000'
         DC    XL4'04030000'
         DC    CL8'STATUS  '
         DC    V(A8ERRO)
         DC    XL4'00000000'
         DC    XL4'00000000'
         DC    XL4'00210000'
         DC    CL8'SUMI    '
         DC    V(A8B00B)
         DC    XL4'206206E2'
         DC    XL4'000000E0'
         DC    XL4'200000E0'
         DC    CL8'SUMJ    '
         DC    V(A8B00B)
         DC    XL4'206206E2'
         DC    XL4'000000E0'
         DC    XL4'200000E0'
         DC    CL8'SUMBI   '
         DC    V(A8B00B)
         DC    XL4'206206E2'
         DC    XL4'000000E0'
         DC    XL4'200000E0'
         DC    CL8'SUMBJ   '
```

```
DC      V(A8B00B)
DC      XL4'206206E2'
DC      XL4'000000E0'
DC      XL4'200000E0'
DC      CL8'SUMMAR  '
DC      V(A8B00B)
DC      XL4'00000000'
DC      XL4'00000000'
DC      XL4'00000000'
DC      CL8'SWAP    '
DC      V(A8SWAP)
DC      XL4'2062061C'
DC      XL4'00000000'
DC      XL4'00010000'
DC      CL8'TAN     '
DC      V(A8B00B)
DC      XL4'20620602'
DC      XL4'000000E0'
DC      XL4'200000E0'
DC      CL8'TEST    '
DC      V(A8TEST)
DC      XL4'0001FF00'
DC      XL4'00000000'
DC      XL4'08050000'
DC      CL8'TITLE   '
DC      V(A8B00B)
DC      XL4'00000000'
DC      XL4'00000000'
DC      XL4'00000000'
DC      CL8'TRAJEC  '
DC      V(A8B00B)
DC      XL4'0060061C'
DC      XL4'ECED0000'
DC      XL4'42060000'
DC      CL8'TRANSL  '
DC      V(A8B00B)
DC      XL4'20622802'
DC      XL4'A4000000'
DC      XL4'40020000'
DC      CL8'TRI     '
DC      V(A8B00B)
DC      XL4'00000A00'
DC      XL4'00000000'
DC      XL4'00000000'
DC      CL8'TXFORM  '
DC      V(A8B00B)
DC      XL4'20600002'
DC      XL4'00000000'
DC      XL4'2004E000'
DC      CL8'TXFRMI  '
DC      V(A8B00B)
DC      XL4'20620602'
DC      XL4'000000E0'
DC      XL4'202000E0'
```

```
DC      CL8'TXFRMJ   '
DC      V(A8B00B)
DC      XL4'20620602'
DC      XL4'000000E0'
DC      XL4'202000E0'
DC      CL8'U,V      '
DC      V(A8UANV)
DC      XL4'00000000'
DC      XL4'00000000'
DC      XL4'30012040'
DC      CL8'VALUE    '
DC      V(A8VALU)
DC      XL4'2183FF00'
DC      XL4'00000000'
DC      XL4'00010000'
DC      CL8'VEARTH   '
DC      V(A8VEAR)
DC      XL4'00000000'
DC      XL4'00000000'
DC      XL4'30012040'
DC      CL8'VGRID    '
DC      V(A8VGRD)
DC      XL4'00000000'
DC      XL4'00000000'
DC      XL4'30012040'
DC      CL8'VLABEL   '
DC      V(A8B00B)
DC      XL4'00108200'
DC      XL4'00000000'
DC      XL4'00480000'
DC      CL8'VORTIC   '
DC      V(A8B00B)
DC      XL4'00000000'
DC      XL4'2E4F0000'
DC      XL4'60006A00'
DC      CL8'WRITE    '
DC      V(A8WRTA)
DC      XL4'00608200'
DC      XL4'00000000'
DC      XL4'00010000'
DC      CL8'XSMOOT   '
DC      V(A8B00B)
DC      XL4'4060061E'
DC      XL4'00E80020'
DC      XL4'6025E020'
DC      CL8'ZERO     '
DC      V(A8B00B)
DC      XL4'0060061C'
DC      XL4'0000E0E0'
DC      XL4'2000E0E0'
DC      CL8'ZONAL    '
DC      V(A8B00B)
DC      XL4'40608202'
DC      XL4'00002000'
```

```
         DC    XL4'20202000'
         DC    CL8'******  '
         DC    V(A8RETN)
         DC    XL4'3FF00000'
         DC    XL4'00000000'
         DC    XL4'025E0000'
         DC    CL8'AND     '
         DC    V(A8B00B)
         DC    XL4'2062061C'
         DC    XL4'E8E80000'
         DC    XL4'200000E8'
         DC    CL8'NOT     '
         DC    V(A8B00B)
         DC    XL4'20600000'
         DC    XL4'E8000000'
         DC    XL4'2000E800'
         DC    CL8'OR      '
         DC    V(A8B00B)
         DC    XL4'2062061C'
         DC    XL4'E8E80000'
         DC    XL4'200000E8'
         DC    CL8'4020ST  '
         DC    V(A8B00B)
         DC    XL4'0001FF00'
         DC    XL4'00000000'
         DC    XL4'40000000'
         DC    CL8'4020EX  '
         DC    V(A8B00B)
         DC    XL4'20600002'
         DC    XL4'00000000'
         DC    XL4'40240000'
         DC    CL8'4020PL  '
         DC    V(A8B00B)
         DC    XL4'2060061E'
         DC    XL4'00E80000'
         DC    XL4'40240000'
         DC    CL8'4020QU  '
         DC    V(A8B00B)
         DC    XL4'20600002'
         DC    XL4'00000000'
         DC    XL4'40240000'
         DC    CL8'4020TA  '
         DC    V(A8B00B)
         DC    XL4'20620602'
         DC    XL4'00000000'
         DC    XL4'400400E8'
         DC    CL8'4020DA  '
         DC    V(A8B00B)
         DC    XL4'22822801'
         DC    XL4'A4000000'
         DC    XL4'40240000'
         DC    CL8'4020TY  '
         DC    V(A8B00B)
         DC    XL4'03F3FF00'
```

```
            DC      XL4'00000000'
            DC      XL4'00000000'
            DC      CL8'4020EN  '
            DC      V(A8B00B)
            DC      XL4'00000000'
            DC      XL4'00000000'
            DC      XL4'00200000'
ENDSTB      EQU     *
            DC      CL8'EXECUT  '
            DC      V(A8EXEC)
            DC      XL4'00000000'
            DC      XL4'00000000'
            DC      XL4'00010000'
            END
```

REFERENCES

(1) World Meteorological Organization,"International Geophysical Year 1957-1958," WMO-No.55.IGY.1, Geneva, 1956, pp. 45-53

(2) Welsh,J.G., "ANAL68 Reference Manual for the UNIVAC 1108," The Travelers Research Center, Hartford, Connecticut, August 1968.

(3) Roos, D., "ICES System Design," The MIT Press, Cambridge, Massachusetts,July 1967.

(4) Garwick,J.V.,"GPL, a Truly General Purpose Language," Communications of the ACM, vol. 11, no. 9, September 1968.

(5) Bobrow,D. and Raphel,B.,"A Comparison of List Processing Computer Languages," Communications of the ACM,vol. 8,no. 4, April 1965.

(6) Rosen, P.,"Electronic Computers: A Historical Survey," Computing Surveys, ACM,vol. 1, no. 1, March 1969, pp 7-36.

(7) Halpern, M., "Toward a General Processor for Programming Languages," Communications of the ACM,vol.11, no. 1, January 1968.

(8) Kolsky,H.G., "Computer Requirements in Meteorology," IBM, Palo Alto Scientific Center, Technical Report No. 320-3210, July 1966.

(9) Frazier,H.M. and Starr, V.P., "An arrangement of Data for Dynamical Studies of the Atmosphere," Prepared for WMO/IUGG Symposium on Meteorological Data Processing,Uccle/Brussels,July 2-5, 1965.

(10) Frazier, H.M.,Sweeton, E.R. and Welsh, J.G., "Data Processing Support to a Program for Observational and Theoretical Studies of Planetary Atmospheres," Progress Report, October 1, 1967, The Travelers Research Center, 7419-339, November 1968.

(11) Programmers Manual, "Output Formats of Station Aerological Data Processed by the IBM 1401/7030 Data Handling Programs," Travelers Research Center, TRC 7453-169, updated April 30,1965.

(12) Seroussi, S.F.,et al, "An Interactive Network of Time Sharing Computers," Paper to be presented at the ACM National Conference in San Francisco, August 1969.